RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation [Vision]

Nan Tang
QCRI, HBKU, Qatar
ntang@hbku.edu.qa

Ju Fan
Renmin University, China
fanj@ruc.edu.cn

Fangyi Li
Renmin University, China
fangyili@ruc.edu.cn

Jianhong Tu
Renmin University, China
tujh@ruc.edu.cn

Xiaoyong Du
Renmin University, China
duyong@ruc.edu.cn

Guoliang Li
Tsinghua University, China
liguoliang@tsinghua.edu.cn

Sam Madden
CSAIL, MIT, USA
madden@csail.mit.edu

Mourad Ouzzani
QCRI, HBKU, Qatar
mouzzani@hbku.edu.qa

ABSTRACT

Can AI help automate human-easy but computer-hard data preparation tasks that burden data scientists, practitioners, and crowd workers? We answer this question by presenting RPT, a denoising autoencoder for *tuple-to-X* models ("*X*" could be tuple, token, label, JSON, and so on). RPT is pre-trained for a *tuple-to-tuple* model by corrupting the input tuple and then learning a model to reconstruct the original tuple. It adopts a Transformer-based neural translation architecture that consists of a bidirectional encoder (similar to BERT) and a left-to-right autoregressive decoder (similar to GPT), leading to a generalization of both BERT and GPT. The pre-trained RPT can already support several common data preparation tasks such as data cleaning, auto-completion and schema matching. Better still, RPT can be fine-tuned on a wide range of data preparation tasks, such as value normalization, data transformation, data annotation, etc. To complement RPT, we also discuss several appealing techniques such as collaborative training and few-shot learning for entity resolution, and few-shot learning and NLP question-answering for information extraction. In addition, we identify a series of research opportunities to advance the field of data preparation.

PVLDB Reference Format:
Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation [Vision]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at http://vldb.org/pvldb/format_vol14.html.

1 INTRODUCTION

Data preparation — including data cleaning [6], data transformation [32], entity resolution [26], information extraction [13], and so forth — is the most time-consuming and least enjoyable work for data scientists [20]. Next, we present several scenarios to better understand these problems.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Q1: *r1*[name, expertise, city] = (Michael Jordan, Machine Learning, **[M]**)
A1: Berkeley
Q2: *r3*[name, affiliation] = (Michael **[M]**, CSAIL MIT)
A2: Cafarella
Q3: *r2*[name, expertise, **[M]**] = (Michael Jordan, Basketball, New York City)
A3: city

(a) Sample Tasks for Value Filling ([M]: value to fill)

	product	company	year	memory	screen
e1	iPhone 10	Apple	2017	64GB	5.8 inchs
e2	iPhone X	Apple Inc	2017	256GB	5.8-inch
e3	iPhone 11	AAPL	2019	128GB	6.1 inches

(b) A Sample Entity Resolution Task

	type	description	label
s1	notebook	2.3GHz 8-Core, 1TB Storage, 8GB memory, 16-inch Retina display	8GB
t1	phone	6.10-inch touchscreen, a resolution of 828x1792 pixels, A14 Bionic processor, and come with 4GB of RAM	4GB

(c) A Sample Information Extraction Task (*s1*: example, *t1*: task)

Figure 1: Motivating Scenarios.

Scenario 1: Data Cleaning. Figure 1(a) Q1 and Q2 show two typical data cleaning problems. (i) *Cell Filling:* Question Q1 asks for the **`city`** for the "Michael Jordan" whose expertise is "Machine Learning". (ii) *Value Filling:* Q2 asks for the last name of someone who works at "CSAIL MIT" with the first name "Michael".

Answering Q1 can help solve a series of problems such as error detection, data repairing, and missing value imputation; and answering Q2 can help auto-completion (*e.g.*, give the answer A2 "Cafarella") and auto-suggestion (*e.g.*, provide a list of candidate names such as {Cafarella, Stonebraker}).

Scenario 2: Attribute Filling for Schema Matching. Figure 1(a) Q3 asks for the attribute name for the value "New York City", *w.r.t.* **`name`** "Michael Jordan" and **`expertise`** "Basketball". Answering this question can help schema matching, a core data integration problem [23], by better aligning attributes from different tables.

Scenario 3: Entity Resolution (ER) Figure 1(b) shows a typical ER task that asks whether e_1, e_2 and e_3 are the "same".

A human with enough knowledge can tell that "iPhone 10" = "iPhone X" ≠ "iPhone 11", "Apple" = "Apple Inc" = "AAPL", and "inches" = "-inch". Hence, one can decide that e_1 and e_2 do not match e_3, and e_1 matches e_2 (if the memory does not matter).

Scenario 4: Information Extraction (IE) Figure 1(c) shows an IE task, which is typically done via crowdsourcing [39]. A requester provides several samples (*e.g.,* s_1) that show what "label" should be extracted, and asks workers to perform similar tasks (*e.g.,* t_1).

A crowd worker needs first to interpret the task by analyzing s_1 (and maybe a few more examples) and concretizes it as "*what is the memory size*". Afterwards, he can perform t_1 by extracting the label 4GB from t_1[`description`] by knowing "RAM" is for memory.

Challenges. Scenarios (1–4) are simple for humans, but are hard for computers. To solve them, computers face the following challenges. (1) *Knowledge:* computers need to have the background knowledge through *understanding* an enormous corpora of tables. (2) *Experience:* computers should be able to learn from prior and various tasks. (3) *Adaptation:* computers should (quickly) adjust to new inputs and new tasks.

Vision. Indeed, these problems have been seen as 'holy grail' problems for the database community for decades [6, 23, 27, 31, 37, 46], but despite thousands of papers on these topics, they still remain unsolved. Recent evidence from the NLP community, where DL-based models and representations have been shown to perform nearly as well as humans on various language understanding and question answering tasks, suggests that a learned approach may be a viable option for these data preparation tasks as well.

Desiderata. The desiderata for AI-powered tools to achieve near-human intelligence for data preparation is summarized as below, in response to Challenges (1–3), respectively. (1) *Deep learning architecture and self-supervised pre-training.* We need a deep learning architecture that can learn from many tables, similar to language models that can learn from a large text corpora. This simulates how humans gain knowledge. Moreover, it should be pre-trained without human provided labels, *i.e.,* self-supervision. (2) *Transfer learning.* The model should be able obtain knowledge from different tasks on different datasets. (3) *Fine-tuning and few-shot learning.* The pre-trained model should allow customization on different downstream applications through fine-tuning. In addition, it should be able to understand a new task from a few examples.

RPT Is *Almost* All You Need. The design of ***R***elational ***P***re-trained ***T***ransformer (RPT) is inspired by recent successes of DL models in NLP. The fundamental questions for relational data understanding on data preparation are: (1) *what is the architecture?* and (2) *what is the surrogate task for pre-training?*

(1) RPT Architecture. Typical choices are encoder-only such as BERT [22], decoder-only such as GPT-3 [11], or encoder-decoder such as BART [44] and T5 [55]. In fact, the encoder-decoder architecture can be considered as a generalization of the encoder-only model (*e.g.,* BERT) and the decoder-only model (*e.g.,* GPT-3). Recent studies from BART and T5 found that encoder-decoder models generally outperform encoder-only or decoder-only language models. Thus, a Transformer-based [63] encoder-decoder model provides more flexibility and can be adapted to a wide range of data preparation tasks, and hence can be used by RPT.

(2) RPT Pre-training. There have been several works on pre-training using tables, such as TAPAS [36], TURL [21], TaBERT [70] and TabFact [15]. However, since most data preparation tasks are in the granularity of tuples, instead of entire tables, we posit that training RPT tuple-by-tuple is more desirable. For the pre-training objectives, most recent studies confirm that *fill-in-the-blank* style denoising objectives (where the model is trained to recover missing pieces in the input) work best; examples include BERT [22], BART [44], and T5 [55] for NLP, and TURL [21] for relational tables.

Contributions. We make the following notable contributions.

- *RPT*: We describe a standard Transformer-based denoising autoencoder architecture to pre-train sequence-to-sequence models for *tuple-to-tuple* training, with new *tuple-aware masking* mechanisms. (Section 2)
- *Fine-tuning RPT:* We discuss a wide range of data preparation tasks that can be supported by fine-tuning RPT. (Section 3)
- *Beyond RPT:* We discuss several appealing techniques that can complement RPT in specific data preparation tasks, *e.g.,* collaborative training and few-shot learning for ER, and few-shot learning and NLP question-answering for IE. (Section 4)

2 RPT

2.1 Architecture

RPT uses a standard sequence-to-sequence (or encoder-decoder) Transformer [63] model, similar to BART [52], as shown in Figure 2.

Encoder. RPT uses a bidirectional encoder (similar to BERT [22]) because it has the advantage of learning to predict the corrupted data bidirectionally, from both the context on the left and the context on the right of the corrupted data. Moreover, it is Transformer-based, which can use self-attention to generate a richer representation of each input token. Hence, a Transformer-based bidirectional encoder is a natural fit for reading tuples where, by definition, the ordering of (attribute name, attribute value) pairs is irrelevant.

Decoder. RPT uses a left-to-right autoregressive decoder (similar to GPT-3 [11]).

2.2 Pre-training RPT

RPT is pre-trained on tuples, for which we just need to corrupt tuples and then optimize a reconstruction loss – the cross-entropy between the model output and the original tuple.

Tuple Tokenization. We represent each tuple as a concatenation of its attribute names and values. For example, tuple t_1 in Figure 1(a) can be tokenized as:

name Michael Jordan expertise Machine Learning city Berkeley

Token Embeddings. Because there is a clear semantic difference between attribute names and values, we can add special tokens, [**A**] before an attribute name and [**V**] before an attribute value. Token embeddings are widely used in NLP tasks, such as the [CLS] (indicating the start) and [SEP] (indicating the next sentence) tokens used by BERT [22]. Hence, we can get a sequence of t_1 with a richer tuple-aware semantics as:

[**A**] name [**V**] Michael Jordan [**A**] expertise [**V**] Machine Learning [**A**] city [**V**] Berkeley

Positional and Column Embeddings. We can add additional meta-data such as positional embeddings (*i.e.,* indicating the token's position in the sequence) and segment embeddings (*e.g.,* adding

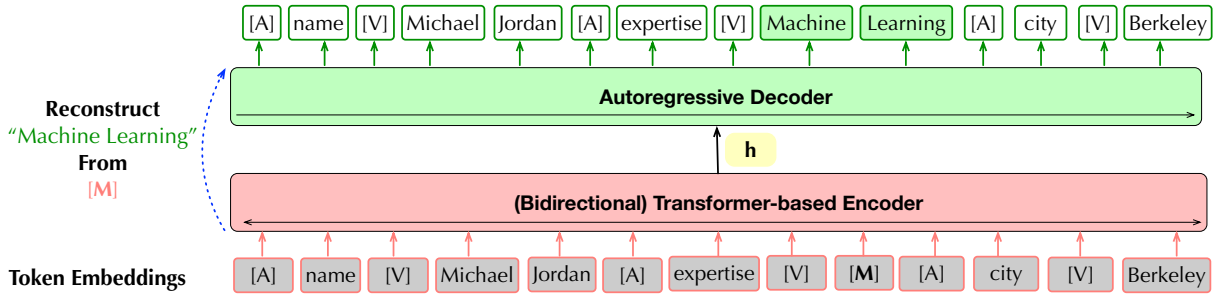

Figure 2: The RPT Architecture.

the same segment embedding to the tokens belonging to the same attribute value), which are inspired by TAPAS [36] for table parsing.

Working Mechanism. Given an input sequence of a tuple with some value to be masked out (*e.g.,* "Machine Learning" in Figure 2) represented by a "mask token" [**M**], along with rich semantic information (*e.g.,* an attribute [**A**] or a value [**V**], which position, and which column), the bidirectional encoder will look at the information before and after the masked token [**M**], learn which token to pay attention to (using Transformer [63]), and generate an intermediate vector representation h. The autoregressive decoder will take h as input, and generate an output sequence, by denoising the masked input [**M**] to be "Machine Learning". By doing so, we can train RPT in an unsupervised fashion, without any human labels.

One difficulty is to predict how many tokens are masked by one [**M**]. Note that, BERT [22] masks each token with one [**M**]; i.e., "Machine Learning" will be masked as [**M**][**M**], which tells explicitly how many tokens are missing. We cannot do the same (masking each token with one [**M**]), because during prediction, we do not know how many tokens are missing. The ability to teach the model to predict the length of the missing tokens masked by one single mask token [**M**] can be achieved by *text infilling* [40], which will be discussed next.

Token Masking. (1) *Attribute Name Masking:* We randomly select attribute names to mask, *e.g.,* **`name`**.

(2) *Entire Attribute Value Masking:* We randomly select entire attribute values to mask, *e.g.,* "Machine Learning" is masked with one [**M**] (see Figure 2), which forces RPT to first predict the number of masked tokens and then predict the tokens.

(3) *Single Attribute Value Masking:* We randomly select a single attribute value (*i.e.,* one token) to mask, *e.g.,* "Jordan".

Note that one possible optimization to the above process is as follows. Instead of giving RPT the full freedom to learn how input tokens attend on each other in the form of an attention matrix [63], we add some explicit rules. For example, (i) an attribute name (*e.g.,* **`name`**) can only attend on the other attribute names (*e.g.,* **`expertise`** and **`city`**) and its associated tokens for attribute values (*e.g.,* "Michael" and "Jordan"), but not other attribute values (*e.g.,* "Berkeley"), and (ii) a token for an attribute value (*e.g.,* "Berkeley") can only attend to all attribute values of all attributes and its attribute name (*i.e.,* **`city`**), but not other attribute names (*e.g.,* **`name`**). TURL [21] also uses this technique, called *visibility matrix*.

2.3 Related Work

Data Cleaning. We categorize prior work on data cleaning into three categories. (i) *Only examine the data at hand.* There are integrity constraints (FDs [9], its extensions CFDs [28] and PFDs [54], denial constraints [18], and rule-based methods [34, 64]), and probabilistic based methods (*e.g.,* HoloClean [57]). They need enough signals or data redundancy from D. Supervised ML based methods (*e.g.,* GDR [68], SCAREd [67], Raha [49] and Baran [48]) learn only from the data at hand, which cannot be generalized to other datasets. (ii) *Use External Reliable Sources:* This includes the of master data [30, 38] or knowledge bases [19, 33]. These methods require experts to define or confirm the matching between the data at hand and the external source, *e.g.,* matching rules for table-table matching [30] or graphical patterns for table-graph matching [19]. (iii) *Human- or crowd-in-the-loop.* When neither (i) nor (ii) type solutions work, a last resort is to fall back on humans to clean the dataset.

Intuitively, with enough signals, data redundancy, reliable external sources with sufficient coverage, and the availability of experts to bootstrap and tune the process, we can leverage (i) and (ii) style solutions. Unfortunately, this is usually not the case in practice [6] — cleaning is frequently of type (iii) with its high human cost. Automating type (iii) solutions is the main motivation of RPT.

Knowledge Bases. There are two ways to encode the knowledge: "explicit knowledge" such as knowledge graphs, or "implicit knowledge" by *memorizing* the knowledge using DL models. In practice, both explicit knowledge graphs and implicit pre-trained DL models have been widely studied in industry and academia. Both directions are important, and RPT belongs to the latter. One drawback is that it is hard to explain, for which explainable AI techniques [24, 42, 58] will play an important role for grounding RPT-like tools.

Relational Table Understanding. TaBERT [70], Tapas [36] and TabFact [15] study the question-answering tasks that involve joint reasoning over both free text and structured tables. They take a natural language utterance as input and produce a structured query (*e.g.,* an SQL query in TaBERT or aggregations in Tapas [36]), or a classification result (*e.g.,* support or refute in TabFact). To this end, they focus on learning a joint representation over textual utterances and tabular data with Transformer models and designing various pre-training tasks to this end.

Closer to this work is TURL [21]. However, RPT differs from TURL in two aspects: (i) TURL employs an encoder-only architecture for learned representations, instead of generating a complicated output, *e.g.,* a tuple or a table. The additional decoder architecture of RPT provides the flexibility of generating sequences in multiple forms. (ii) TURL has to be used with a KB to extract values. For example, for cell infilling, TURL uses a pre-trained model (1.2GB) to

generate a representation, which has to be linked to the KB (*i.e.,* a collection of web tables, 4.6GB) to get the actual value. RPT (1.6GB in our experiment) does not need such a KB to fill missing values.

2.4 Opportunities

RPT naturally supports several common data preparation tasks, *e.g.,* error detection, data repairing, auto-completion, auto-suggestion, and schema matching. Yet there are also many opportunities.

(O1) *Hybrid Solutions.* While RPT does not differentiate between categorical or numeric data during pre-training, it works better for categorical data (*i.e.,* human-easy). A promising direction is to combine RPT with other (quantitative) data cleaning methods [53] from a rich set of (a-b) type data cleaning solutions.

(O2) *Dirty Data.* Many tables are dirty. Pre-training RPT on these dirty tables may yield a biased result. Currently, we learn directly from dirty tables, by assuming that the frequency of correct values is higher than the frequency of wrong values. There are several open problems. First, we would like to provide some guarantee of model robustness while still learning from dirty data. Second, a cleaned version of training data that can be used as a benchmark is highly desired, similar to the Colossal Clean Crawled Corpus (C4) for Text-To-Text-Transfer-Transformer (T5) [55].

(O3) *An AI-assisted Tool with Human-in-the-loop.* Achieving high accuracy in diverse data preparation tasks and domains is still a challenge for RPT; it would require substantial in-domain training data. Hence, a practical usage of RPT is to use it as an AI-assisted tool that can suggest meaningful results in many human-in-the-loop tasks, which can guide users and thus reduce human cost.

2.5 Preliminary Result

We have conducted preliminary experiments to show that RPT can reconstruct the masked token(s) in tuples. Our baseline is BART [52], which is pre-trained with a large corpus of text, including from the product domain. Because BART and RPT have the same architecture (Figure 2), we can use the parameters pre-trained by BART, instead of a random initialization. We used tables about products, including Abt-Buy [1] and Walmart-Amazon [5]. Note that these two tables are naturally dirty.

For testing, we used Amazon-Google [2], the tables that were not seen by BART or RPT. We masked attribute values and asked BART and RPT to predict the original values. Table 1 shows some results, where **[M]** means that the value is masked out, column Truth is the ground truth, and columns RPT and BART provide the results predicted by each, respectively. Tuples 1-2 involve predicting missing prices, where RPT gives close predictions but BART does not. Tuples 3-4 involve predicting missing manufacturers; RPT-C provides good predictions. Tuple 5 involves predicting a missing title, and RPT provides a partially correct prediction.

This preliminary experiment shows that RPT pre-trained on tables can learn structural data values from tables better than directly using a pre-trained language model (*e.g.,* BART), which is not customized for relational data. The main reason is that, by pre-training on the product tables, RPT can better learn dependency among columns, and thus is more capable of predicting missing values. Of course, RPT sometimes makes wrong predictions, but for those cases, BART also fails. Our belief is that these preliminary results are suggestive enough of the effectiveness of the approach that it merits significant additional investigation.

title	manufacturer	price	Truth	RPT-C	BART
instant home design (jewel case)	topics entertainment	[M]	9.99	9	Topics
disney's 1st & 2nd grade bundle ...	disney	[M]	14.99	19	Dis
adobe after effects professional 6.5 ...	[M]	499.99	adobe	adobe	$1.99
stomp inc recover lost data 2005	[M]	39.95	stomp inc	stomp	39.95
[M]	write brothers	269.99	write brothers dramatica ...	write brothers	1.99

Table 1: Compare RPT with BART (yellow: masked values; green: (partially) correct; pink: wrong).

Limitations. RPT faces similar limitations that pre-trained language models (LMs) face. (1) *Numeric values:* numeric values are usually mapped into unknown tokens causing the model to fail on tasks that require precise prediction on numeric values. (2) *Max sequence length:* restricted by the GPU memory size, most pre-trained LMs are limited by the sequence length, thus data preparation on wide tables may require additional optimization. (3) *Not fully reliable.* Similar to GPT-3, a generative model cannot be fully trusted. One way to combat this is to treat it as an AI-assistant with a human-in-the-loop, as discussed in Section 2.4 Opportunities (O3).

3 FINE-TUNING RPT

The encoder-decoder architecture of RPT (pre-trained on *tuple-to-tuple*) provides the flexibility to be fine-tuned for different downstream data preparation tasks (*i.e., tuple-to-X*).

Value Normalization. Because RPT has an autoregressive decoder, it can be directly fine-tuned for sequence generation tasks such as value normalization (*e.g.,* "Mike Jordan, 9 ST, Berkeley" → "Mike Jordan, 9th Street, Berkeley"). The encoder takes the input value as a sequence and the decoder generates the output autoregressively. In addition, normalizing "Mike" to "Michael" or "Sam" to "Samuel" can be fine-tuned as a neural name translation [62] task.

Data Transformation. Similarly to what is described above, RPT can be fine-tuned for transformation of data from one format (*e.g.,* a tuple) to another format (*e.g.,* JSON or XML), where the decoder will autoregressively serialize the output in the target format.

Data Annotation. Given a tuple, data annotation requires adding a label (*e.g.,* a classification task). We can use the final hidden state of the final decoder token to fine-tune a multi-class linear classifier.
Information Extraction (IE). Given a tuple, IE extracts a span or multiple spans of relevant text, which can be done by fine-tuning the decoder to produce the (start, end) pairs of spans.

Learned Tuple Representation for Entity Resolution. The embeddings of entities have been used in entity resolution for both

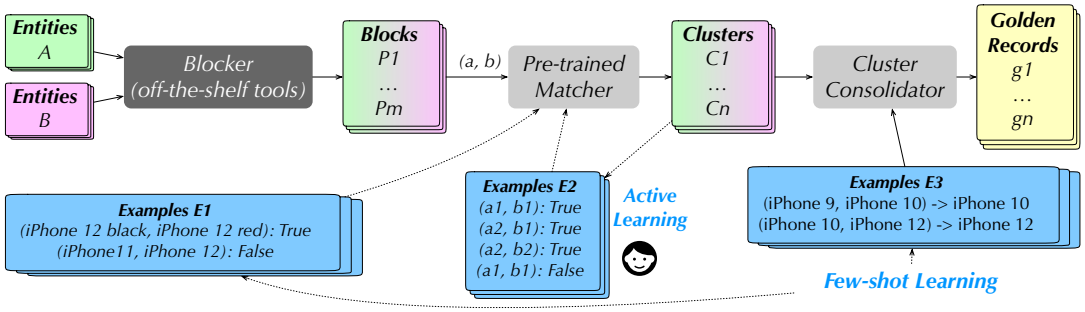

Figure 3: Collaborative Learning and Few-shot Learning for Entity Resolution.

blocking [25] and entity matching [25, 50]. A typical trick is to do cross-tuple training (or contrastive learning [14]), via Siamese NNs [16], such that similar entities have similar embeddings. Similarly, the encoder of RPT can be fine-tuned in Siamese NNs for learned representations *w.r.t.* entity resolution.

4 BEYOND RPT

In Section 3, we discussed how RPT can be fine-tuned for several data preparation tasks. However, not all human-easy tasks can be automated. In practice, many objectives are subjective and many tasks are ambiguous or underspecified. Here we explore other, related, techniques that can help on specific tasks.

4.1 Entity Resolution

Given two sets of entities, A and B, an end-to-end of ***entity resolution pipeline*** (Figure 3) is: (1) find duplicated entity pairs ($a \in A, b \in B$) (blocking to improve efficiency); (2) merge them into clusters, typically through transitive closure, and (3) consolidate each cluster into one entity.

Blocking. There is a rich literature on automatic blocking for ER (see [51] for a survey). There are also DL-based methods [25, 45, 65] to generate blocks. These prior works are automatic and already work well, hence will not be covered in this paper.

Matcher. The state-of-the-art matchers are all ML based *e.g.,* random forests (*e.g.,* Magellan [43]), or DL based (*e.g.,* DeepMatcher [50] and DeepER [25]). Recent works [12, 45] also study to leverage pre-trained LM models for generating entity representations.

Consolidator. There have been rule-based methods [29] and learning-based approach [35] for entity consolidation – both need either significant human involvement or a large amount of training data.

Vision and Opportunities. The Matcher and Consolidator should be able to perform effectively through pre-trained models (*i.e.,* to obtain knowledge) and a few examples (*i.e.,* to interpret the task).

However, this pipeline cannot be fully automated, because some judgments are *objective*, *e.g.,* "iPhone 10", "iPhone ten", and "iPhone X" are the same, while some others are *subjective*, *e.g.,* whether "iPhone 12 red" matches "iPhone 12 black" is user dependent.

Our intuition is that the objective criteria can be pre-trained (such as "iPhone 10" matches "iPhone X" and "Mike" matches "Michael"), but the subjective criteria need task-specific samples, for both the Matcher and the Consolidator (Figure 3), which could be achieved by getting a few examples from humans.

We identify two major opportunities for the entire ER pipeline.

(O1) *Collaborative learning* or *Federated Learning (FL) [10, 69].* This is to learn the "objective" criteria for the Matcher. Note that there are many public and private ER benchmarks, which share common domains. It is promising to collaboratively train one Matcher, and the knowledge can be learned and transferred from one dataset to another dataset. Better still, this can be done securely [47], without data sharing. Note that, there have been transfer learning techniques on ER [41, 60] to show an early success on this thread.

We believe that we should build a platform collaboratively for ER, with a pre-trained model M for each domain. Anyone who wants to benefit from M can download M, retrain using his/her data to get a M_1, and send back an update of parameters $\Delta_1 = M_1 - M$, and the platform will merge the model update with M, from multiple users [10]. Because different entities may have different schemas, we use a pre-trained model such as BERT to be schema-agnostic.

(O2) *Few-shot Learning.* This is to learn the "subjective" criteria, for Matcher and Consolidator, through a human-in-the-loop approach. The goal is to infer a better specified task from a few examples, *e.g.,* using Pattern-Exploiting Training [59].

[Matcher.] Consider E_1 in Fig. 3 that contains two user provided examples and we want to automatically generate a clearer task for workers, *e.g.,* "color does not matter but model matters". We can design two templates like (T1) "True: if $\boxed{a}$ and $\boxed{b}$ have the same $[\mathbf{M}]_1$" and (T2) "False: if $\boxed{a}$ and $\boxed{b}$ have different $[\mathbf{M}]_2$". By replacing the first matching pair in E_1 to template (T1), we can infer a pattern "model" or "series" (but not "color") for $[\mathbf{M}]_1$. Similarly, by using the second un-matching pair in E_2 to template (T2), we can infer a pattern "model" or "series" for $[\mathbf{M}]_2$.

Moreover, when merging matching entities into clusters based on transitive closure, conflict may be automatically detected within clusters (*e.g.,* E_2 in Fig. 3); such conflicts can be resolved by the users through active learning. Note that, doing active learning from conflicting predictions is different from traditional active learning methods on ER that use confusing/informative entity pairs [7, 17].

[Consolidator.] Consider E_3 with two examples, "iPhone 10 is more preferred than iPhone 9", and "iPhone 12 is more preferred than iPhone 10". We can use them to make the task clearer by asking questions "iPhone 10 is [**M**] than iPhone 9" and "iPhone 12 is [**M**] than iPhone 10", and enforce a language model to fill the two masked tokens with the same value, which might be "newer".

Method	Abt-Buy		Amazon-Google	
	F1 score	# labels	F1 score	# labels
Collaborative Training (CT)	0.72	0	0.53	0
ZeroER	0.52	0	0.48	0
DeepMatcher	0.63	7689	0.69	9167
Ditto	0.71	1500	0.50	1000

Table 2: Comparison with the State-of-the-art.

Another powerful method related to few-shot learning is meta-learning (or learning to learn fast) [61], with the main goal to learn new concepts and skills fast with a few examples.

Preliminary Results. We have conducted some preliminary experiments on the "product" domain for the Matcher, because this domain has a rich collection of ER benchmarks, and it is known to be hard ER cases because they are text-rich. Specifically, we use five well-known benchmarks. (D1) Abt-Buy [1], (D2) Amazon-Google [2], (D3) Walmart-Amazon [5], (D4) iTunes-Amazon [3], and (D5) SIGMOD 2020 programming contest [4] (we took 1000/8000 matching/unmatching pairs). These datasets are not cleaned.

Specifically, when testing on D1, we train with D2–D5, and when testing on D2, we train with D1, D3–D5. We only tested on D1 and D2, so we can directly compare with ZeroER [66] (without any training data) and DeepMatcher [50] (trained with hundreds/thousands of examples), where the F1 scores and the number of labels are reported from their original papers. We also compare with Ditto [45]. We progressively add the number of labeled data to fine-tune Ditto until its F1 is very close to that of collaborative training (CT).

We also note that the paper [12] is quite similar to Ditto. First, both studies use the same strategy to model a record pair as a sequence and fine-tune a pre-trained model to output the matching results. Second, [12] compares different pre-trained LM models for entity resolution and reports that RoBERTa [25] achieves the best performance. Similarly, the latest version of Ditto [45] also uses RoBERTa as its pre-trained model. Third, both studies achieve similar results in the experiments. For example, on the Abt-Buy dataset, [12] and Ditto achieve 0.91 and 0.90 on F1 score respectively.

Table 2 shows that CT outperforms ZeroER and is comparable with DeepMatcher that was trained with 1000+ examples. Moreover, CT uses zero examples from the test ER dataset to achieve the performance of Ditto, which is trained by fine-tuning a pre-trained model with 1000+ examples. This result verifies the opportunity (O1) that it is promising to collaboratively train a Matcher to decide whether two entities (even in different schemas) match or not.

4.2 Information Extraction

Information Extraction (IE) is the process of retrieving specific information from unstructured (*e.g.,* text), or (semi-)structured (*e.g.,* relational) data. We consider a simple IE problem: given a text or text-rich tuple t, it is to extract a span (*i.e.,* a continuous sequence of tokens) from t, denoted by $\mathbf{I}(t)$. See Figure 1(c) for a sample IE task. Although simple, the above definition is general enough to cover a wide range of IE problems.

Connection with Question-Answering in NLP. There is a natural connection between the IE problem and a typical question-answering problem in NLP. For question answering, there is an input question such as "Q: where do water droplets collide with ice crystals to form precipitation", and an input paragraph "P: ...

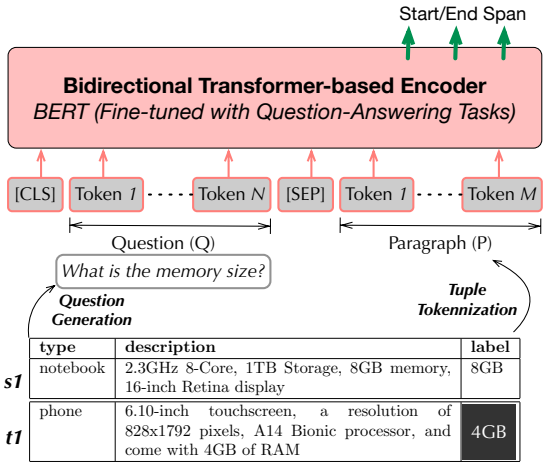

Figure 4: Connecting IE with NLP Question-Answering.

Precipitation forms as smaller droplets coalesce via collision with other rain drop or ice crystals within a cloud. ...". The task is to find a span *e.g.,* within a cloud of paragraph P to answer the question Q. There are many NLP question-answering benchmarks, such as SQuAD [56] where AI has outperformed human performance [71].

We have discussed how to fine-tune RPT for IE in Section 3. Alternatively, we can leverage other mature solutions from well-known NLP tasks such as NLP query-answering. As shown in Figure 4, given a query Q, and a paragraph P, a pre-trained model fine-tuned using question-answering benchmarks can provide a span, *i.e.,* (start, end) positions, as output.

We can tokenize a tuple t as a paragraph P (Section 2). The remaining problem is to generate the question Q. We can have a question template such as "what is the [**M**]", where the [**M**] can be instantiated with one-shot learning (*e.g.,* the label of s_1) via *e.g.,* PET [59], which gives "what is the memory size" as the question Q.

Opportunities. (O1) Connect more DB-related IE tasks to well-studied NLP tasks, so as to obtain pre-trained knowledge (*e.g.,* NeruON [8] uses a seq-to-seq model to extract tuples from question-answer pairs). (O2) Currently, many IE tasks are performed by crowd workers (or crowd-in-the-loop). Instead of fully replacing these crowd workers, we are studying how to train multiple RPT-I models as AI-workers, and mix the AI-workers and crowd workers to reduce the total cost of a crowdsourcing task.

5 CALL TO ARMS

We have presented our vision for democratizing data preparation, including concrete steps towards this goal: RPT, fine-tuning RPT, and other appealing techniques. Several recent successes (*e.g.,* TURL [21], Ditto [45] and NeurON [8]) have shed some light on this direction. Our preliminary results, along with these related papers suggest that learning-based approaches have the potential to outperform more traditional methods, much as they have revolutionized NLP. However, the data preparation field is vast, the problems are diverse and much work remains to be done. In particular, a major obstacle to advance all the above topics is the limited availability of real-world benchmarks, *e.g.,* C4 for T5 [55]. Now is the time for the data preparation and larger database communities to come together to explore the potential of these new techniques.

REFERENCES

[1] Abt-buy. https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md#abt-buy.

[2] Amazon-google. https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md#amazon-google.

[3] Itunes-amazon. https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md#itunes-amazon.

[4] sigmod-2020-contest. http://www.inf.uniroma3.it/db/sigmod2020contest/task.html.

[5] Walmart-amazon. https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md#walmart-amazon.

[6] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.*, 9(12):993–1004, 2016.

[7] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi. Active sampling for entity matching with guarantees. *ACM Trans. Knowl. Discov. Data*, 7(3):12:1–12:24, 2013.

[8] N. Bhutani, Y. Suhara, W. Tan, A. Y. Halevy, and H. V. Jagadish. Open information extraction from question-answer pairs. In J. Burstein, C. Doran, and T. Solorio, editors, *NAACL-HLT*, pages 2294–2305, 2019.

[9] P. Bohannon, M. Flaster, W. Fan, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In F. Özcan, editor, *SIGMOD*, pages 143–154, 2005.

[10] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards federated learning at scale: System design. In *MLSys*, 2019.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

[12] U. Brunner and K. Stockinger. Entity matching with transformer architectures - A step forward in data integration. In *EDBT*, pages 463–473, 2020.

[13] M. J. Cafarella, J. Madhavan, and A. Y. Halevy. Web-scale extraction of structured data. *SIGMOD Rec.*, 37(4):55–61, 2008.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[15] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang. Tabfact: A large-scale dataset for table-based fact verification. *CoRR*, abs/1909.02164, 2019.

[16] D. Chicco. Siamese neural networks: An overview. In *Artificial Neural Networks - Third Edition*, volume 2190 of *Methods in Molecular Biology*, pages 73–94. 2021.

[17] V. Christen, P. Christen, and E. Rahm. Informativeness-based active learning for entity resolution. In P. Cellier and K. Driessens, editors, *ECML PKDD*, volume 1168, pages 125–141, 2019.

[18] X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. *Proc. VLDB Endow.*, 6(13):1498–1509, 2013.

[19] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In T. K. Sellis, S. B. Davidson, and Z. G. Ives, editors, *SIGMOD*, pages 1247–1261, 2015.

[20] D. Deng, R. C. Fernandez, Z. Abedjan, S. Wang, M. Stonebraker, A. K. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, and N. Tang. The data civilizer system. In *CIDR*, 2017.

[21] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. TURL: table understanding through representation learning. *Proc. VLDB Endow.*, 2020.

[22] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *NAACL-HLT*, pages 4171–4186.

[23] A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.

[24] F. Doshi-Velez and B. Kim. A roadmap for a rigorous science of interpretability. *CoRR*, abs/1702.08608, 2017.

[25] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467, 2018.

[26] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[27] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2012.

[28] W. Fan, F. Geerts, X. Jia, and A. Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.*, 33(2):6:1–6:48, 2008.

[29] W. Fan, F. Geerts, N. Tang, and W. Yu. Inferring data currency and consistency for conflict resolution. In C. S. Jensen, C. M. Jermaine, and X. Zhou, editors, *ICDE*, pages 470–481, 2013.

[30] W. Fan, J. Li, S. Ma, N. Tang, and W. Yu. Towards certain fixes with editing rules and master data. *Proc. VLDB Endow.*, 3(1):173–184, 2010.

[31] B. Golshan, A. Y. Halevy, G. A. Mihaila, and W. Tan. Data integration: After the teenage years. In E. Sallinger, J. V. den Bussche, and F. Geerts, editors, *PODS*, pages 101–106. ACM, 2017.

[32] M. Hameed and F. Naumann. Data preparation: A survey of commercial tools. *SIGMOD Rec.*, 49(3):18–29, 2020.

[33] S. Hao, N. Tang, G. Li, and J. Li. Cleaning relations using knowledge bases. In *ICDE*, pages 933–944, 2017.

[34] J. He, E. Veltri, D. Santoro, G. Li, G. Mecca, P. Papotti, and N. Tang. Interactive and deterministic data cleaning. In *SIGMOD*, pages 893–907, 2016.

[35] A. Heidari, G. Michalopoulos, S. Kushagra, I. F. Ilyas, and T. Rekatsinas. Record fusion: A learning approach. *CoRR*, abs/2006.10208, 2020.

[36] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *ACL*, pages 4320–4333, 2020.

[37] I. F. Ilyas and X. Chu. *Data Cleaning*. ACM, 2019.

[38] M. Interlandi and N. Tang. Proof positive and negative in data cleaning. In *ICDE*, pages 18–29, 2015.

[39] A. Jain, A. D. Sarma, A. G. Parameswaran, and J. Widom. Understanding workers, developing effective tasks, and enhancing marketplace dynamics: A study of a large crowdsourcing marketplace. *Proc. VLDB Endow.*, 10(7):829–840, 2017.

[40] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. Spanbert: Improving pre-training by representing and predicting spans. *Trans. Assoc. Comput. Linguistics*, 8:64–77, 2020.

[41] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa. Low-resource deep entity resolution with transfer and active learning. In A. Korhonen, D. R. Traum, and L. Màrquez, editors, *ACL*, pages 5851–5861, 2019.

[42] B. Kim and F. Doshi-Velez. Interpretable machine learning: The fuss, the concrete and the questions. In *ICML Tutorial*, 2017.

[43] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra. Magellan: Toward building entity matching management systems. *Proc. VLDB Endow.*, 9(12):1197–1208, 2016.

[44] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, 2020.

[45] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*

[46] B. Liu, L. Chiticariu, V. Chu, H. V. Jagadish, and F. Reiss. Automatic rule refinement for information extraction. *Proc. VLDB Endow.*, 3(1):588–597, 2010.

[47] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang. A secure federated transfer learning framework. *IEEE Intell. Syst.*, 35(4):70–82, 2020.

[48] M. Mahdavi and Z. Abedjan. Baran: Effective error correction via a unified context representation and transfer learning. *Proc. VLDB Endow.*, 13(11):1948–1961, 2020.

[49] M. Mahdavi, Z. Abedjan, R. C. Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Raha: A configuration-free error detection system. In *SIGMOD*, pages 865–882, 2019.

[50] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In G. Das, C. M. Jermaine, and P. A. Bernstein, editors, *SIGMOD*, pages 19–34, 2018.

[51] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Comput. Surv.*, 53(2):31:1–31:42, 2020.

[52] O. Press, N. A. Smith, and O. Levy. Improving transformer models by reordering their sublayers. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *ACL*, pages 2996–3005.

[53] N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. Combining quantitative and logical data cleaning. *Proc. VLDB Endow.*, 9(4):300–311, 2015.

[54] A. A. Qahtan, N. Tang, M. Ouzzani, Y. Cao, and M. Stonebraker. Pattern functional dependencies for data cleaning. *Proc. VLDB Endow.*, 13(5):684–697, 2020.

[55] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[56] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. In J. Su, X. Carreras, and K. Duh, editors, *EMNLP*, pages 2383–2392, 2016.

[57] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, 2017.

[58] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144, 2016.

[59] T. Schick and H. Schütze. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676, 2020.

[60] S. Thirumuruganathan, S. A. P. Parambath, M. Ouzzani, N. Tang, and S. R. Joty. Reuse and adaptation for entity resolution through transfer learning. *CoRR*, abs/1809.11084, 2018.

[61] S. Thrun and L. Y. Pratt. Learning to learn: Introduction and overview. In S. Thrun and L. Y. Pratt, editors, *Learning to Learn*, pages 3–17. Springer, 1998.
[62] A. Ugawa, A. Tamura, T. Ninomiya, H. Takamura, and M. Okumura. Neural machine translation incorporating named entity. In *COLING*, pages 3240–3250, 2018.
[63] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.
[64] J. Wang and N. Tang. Towards dependable data repairing with fixing rules. In *SIGMOD*, pages 457–468. ACM, 2014.
[65] Z. Wang, B. Sisman, H. Wei, X. L. Dong, and S. Ji. Cordel: A contrastive deep learning approach for entity linkage. 2020.
[66] R. Wu, S. Chaba, S. Sawlani, X. Chu, and S. Thirumuruganathan. Zeroer: Entity resolution using zero labeled examples. In *SIGMOD*, pages 1149–1164, 2020.
[67] M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In K. A. Ross, D. Srivastava, and D. Papadias, editors, *SIGMOD*, pages 553–564. ACM, 2013.
[68] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *Proc. VLDB Endow.*, 4(5):279–289, 2011.
[69] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.
[70] P. Yin, G. Neubig, W. Yih, and S. Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *ACL*, pages 8413–8426, 2020.
[71] Z. Zhang, J. Yang, and H. Zhao. Retrospective reader for machine reading comprehension, 2020.